

REMARKS/ARGUMENTS

Claims 1-31 and 43-48 are pending, claims 1-5, 8, 9, 15, 17, 21, 22, 27-29, and 43-48 are amended.

The undersigned attorney thanks the Examiner for his time for the telephonic interview of June 26, 2008.

Claims 1-5, 7-18, 20-30 and 43-48 are rejected under 35 U.S.C. 102(e) as being anticipated by Jorapur (U.S. 7,299,382). Claims 6, 19, and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Jorapur in view of Man et al. (U.S. 6,625,760). Applicant submits that all of the claims currently pending in this application are patentably distinguishable over the cited references for the following reasons, and reconsideration and allowance of this application are respectfully requested.

Amended independent claims 1 and 14 include, among other elements, "executing a plurality of software verification programs to verify the computer software, wherein each of the plurality of software verification tools corresponds to a respective lifecycle phase of the computer software, and automatically generates one or more test cases from the source code of the computer software," "executing a first software verification program corresponding to a first lifecycle phase of the computer software, wherein the first software verification program automatically generates one or more test cases from the source code of the computer software," and "executing a second software verification program corresponding to a second lifecycle phase of the computer software different from the first lifecycle phase." Jorapur does not teach the above elements.

Regarding the limitation of "executing a plurality of software verification programs to verify the computer software, wherein each of the plurality of software verification tools . . . automatically generates one or more test cases from the source code " the examiner cites to col. 4:55-65; col. 6:52-60; and col. 9:45-55 of Jorapur as teaching this limitation. Applicant respectfully disagrees. The above cited passages of Jorapur simply teaches different tests (e.g., test cases) that may be generated in one or more blocks corresponding to one or more parts of the

program under test. These tests 302 are generated by a single test generator 301. "The test generator may be a script, an application, or other element configured to generate tests for software. The test generator 301 may return one or more tests 302 corresponding to the application 300. In one embodiment, the test generator 301 produces tests 302 corresponding to modules that are part of the application 300. For example, the test generator 301 may access one or more modules of the application 300 and generate tests 302 including test code configured to interact with at least one of the modules. (Col. 6, lines 41-52, and FIG. 3, emphasis added.).

The Examiner states that "because all is required by this limitation is that the fact that the software verification tools tests the software errors." (Final Office action, page 2, middle of the last paragraph.). However, Applicant respectfully submit that the limitation requires that "each of the plurality of software verification tools . . . automatically generates one or more test cases from the source code." However, it is clear from Jorapur's disclosure (e.g., see the above cited text) that the individual tests 302 do not and can not "automatically generate one or more test cases." Rather, at best, they may constitute the "generated test cases" themselves, generated by the single test generator 302.

Furthermore, neither these individual tests 302, nor the single test generator 301 of Jorapur can be constituted as the claimed "plurality of software verification programs," for example, Jtest™, C++Test™, WebKing™, SOAPtest™, CodeWizard™, DataRecon™, SOAPbox™, and WebBox™, from Parasoft Corp. (page 14, lines 6-8). For example, CodeWizard™ tool (program) uses Source Code Analysis Technology to parse code (e.g., C and C++ code) looking for specific coding patterns that compilers do not detect. (Page 23, lines 18-20). Jtest™ tool (program) automatically tests code construction (white-box testing), tests code functionality (black-box testing), and maintains code integrity (regression testing). (page 10, lines 1-3). SOAPtest™ tool (program) test web services including simple object access protocol (SOAP), etc.

Additionally, one skilled in the art would readily understand that a software verification program or tool is different than simply a test that may be executed by the program (or the tool), among many other tests. In fact, Wikipedia defines a software tool as "a program or application

that software developers use to create, debug, maintain, or otherwise support other programs and applications." (See, wikipedia.org).

As a result, the generated tests 302 (test cases) of Jorapur can not be constituted as the claimed verification programs and therefore, Jorapur does not disclose "executing a plurality of software verification programs to verify the computer software."

With respect to the limitation of "wherein each of the plurality of software verification programs relates to a respective lifecycle phase of the computer software," the Examiner responds that Jorapur discloses "automated tests may cover the life cycle of an application not simply during execution but from deployment to undeployment to catch more potential errors." (Final Office action, page 3, last paragraph.). However, the fact that the tests may cover the lifecycle of an application, does not mean that the "software verification programs relates to a respective lifecycle phase." Jorapur in deed teaches that these generated tests 302 "may reveal different problems during testing" and "may cover the life cycle of an application," however, this teaching does not amount to disclosing "each of the plurality of software verification programs relates to a respective lifecycle phase of the computer software," such as, Design phase, Development phase, Deploy phase, and Manage phase, as shown in FIG. 1A of the specification.

In any case, claims 1 and 14 are amended to include the extra limitations of "executing a first software verification program relating to a first lifecycle phase of the computer software, wherein the first software verification program automatically generates one or more test cases from the source code of the computer software," and "executing a second software verification program relating to a second lifecycle phase of the computer software different from the first lifecycle phase, wherein the second software verification program automatically generates one or more test cases from the source code of the computer software," to clarify the above-mentioned distinctions. It is clear that Jorapur does not teach the above additional limitations.

Consequently, for the above reasons, Jorapur does not teach all elements of amended claims 1 and 14 and therefore, claims 1 and 14 are not anticipated by Jorapur .

Independent claims 27 and 45 include, among other elements, "providing a plurality of software verification programs each of the plurality of software verification programs related to a

respective lifecycle phase of the computer software," "determining what phase of the lifecycle the error was introduced, based on analyzing the known error," and "executing the plurality of software verification programs to verify the class of the known error is detected in a respective lifecycle phase of computer software." Jorapur does not teach the above elements.

First, regarding the element of "providing a plurality of software verification programs each of the plurality of software verification programs related to a respective lifecycle phase of the computer software," as explained above this limitation is not disclosed by Jorapur.

Second, the element of "determining what phase of the lifecycle the error was introduced, based on analyzing the known error" is also not disclosed by Jorapur. There is no disclosure in Jorapur regarding "determining what phase of the lifecycle the error was introduced." Additionally, as discussed above, with respect to claims 1 and 14, the individual tests 302 of Jorapur are not equivalent to the claimed verification programs, and do not relate to different lifecycle phases.

Third, with respect to "executing the plurality of software verification programs to verify the class of the known error is detected in a respective lifecycle phase of computer software" Jorapur does not teach any "class of known error," rather, Jorapur is directed to a plurality of individual tests 302 for detecting individual errors (not class of errors).

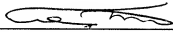
As a result, for at least each of the above three reasons, Jorapur does not teach all elements of claims 27 and 45 and thus, claims 27 and 45 are not anticipated by Jorapur either.

Dependent claims 2-13, 15-26, 28-31, 43-44, and 46-48 are dependent from allowable independent claims 1, 14, 27, and 45, respectively and therefore include all the limitations of their base claims and additional limitations therein. Accordingly, these claims are also allowable over the cited references, as being dependent from an allowable independent claim and for the additional limitations they include therein.

Appln No. 10/613,166
Amdt date July 3, 2008
Reply to Office action of May 14, 2008

In view of the foregoing remarks, it is respectfully submitted that this application is now in condition for allowance, and accordingly, reconsideration and allowance are respectfully requested.

Respectfully submitted,
CHRISTIE, PARKER & HALE, LLP

By 
Raymond R. Tabandeh
Reg. No. 43,945
626/795-9900

RRT/clv

CLV PAS802270.1-*07/3/08 3:34 PM